

[zurück](#)

# Load Balancing - Grundlagen (L4 & L7)

Load Balancing verteilt Anfragen (Traffic) auf mehrere Server, um Verfügbarkeit, Leistung und Ausfallsicherheit zu erhöhen.

Typische Einsatzgebiete:

- Webseiten (z. B. [www.example.com](http://www.example.com))
- APIs & Microservices
- Docker / Kubernetes
- Unternehmensanwendungen
- Datenbanken (Read-Replicas)
- E-Mail-Cluster
- VPN-Gateways

## Warum Load Balancing?

- höhere Verfügbarkeit (kein Single Point of Failure)
- bessere Performance
- mehr Kapazität
- Wartung ohne Downtime
- DDoS-Abmilderung

## Zwei Hauptarten

- **Layer 4 Load Balancing** (Transport Layer)
- **Layer 7 Load Balancing** (Application Layer)

---

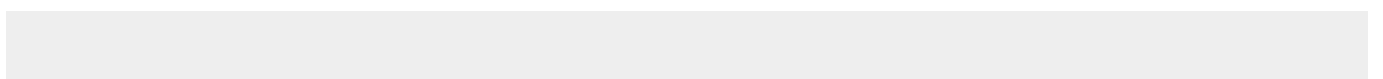
# 1. Layer 4 Load Balancing (Transport-Layer)

L4 arbeitet auf Basis von:

- IP-Adresse
- Port
- TCP/UDP

Der Load Balancer weiß NICHT, was im HTTP/TLS passiert – er verteilt nur Verbindungen.

ASCII:



```
Client → LB (L4) → Server1
                → Server2
                → Server3
```

## Merkmale

- sehr schnell
- arbeitet mit TCP/UDP direkt
- keine Inhalteinsicht
- gut für VPN, Datenbanken, SMTP, Spiele-Server

## Beispiele

- HAProxy (L4/L7)
- Linux IPVS (z. B. keepalived)
- LVS Load Balancer
- AWS NLB (Network Load Balancer)

## Typische L4-Methoden

- Round Robin
  - Least Connections
  - \* Source IP Hash
  - \* Weighted Scheduling

—

# 2. Layer 7 Load Balancing (Application-Layer)

L7 „versteht“, was im HTTP/HTTPS passiert, und kann Inhalte analysieren.

Kann abhängig machen von:

- URLs
- Headern
- Cookies
- Benutzeragenten
- Sessions
- APIs

ASCII:

```
Client → LB (L7)
├── /api → Server1
├── /images → Server2
└── /admin → Server3
```

## Merkmale

- sehr flexibel
  - Routing nach Inhalt
    - \* SSL/TLS-Terminierung möglich
    - \* ideal für Webservices

## Beispiele

- Nginx
  - Traefik
    - \* HAProxy (kann L7)
    - \* Envoy
    - \* F5 Big-IP
    - \* AWS ALB (Application Load Balancer)

## Vorteile

- intelligenter Trafficfluss
  - HTTPS-Aufbrechen & Analyse
    - \* API-Gateway-Funktionen

## Nachteile

- mehr CPU-Bedarf
  - komplexer
    - \* TLS-Handling muss sauber sein

—

# 3. Load Balancing-Methoden

## Round Robin

Jede Anfrage nacheinander an anderen Server.

```
1 → S1
2 → S2
3 → S3
4 → S1
```

## Least Connections

Der Server mit den wenigsten aktiven Verbindungen bekommt die nächste Anfrage.

## Source IP Hash

Gleiche Quell-IP → immer gleicher Server (Sessionstickiness).

## Weighted Round Robin

Server mit mehr Leistung bekommen mehr Anfragen.

---

# 4. Health Checks

Ein Load Balancer prüft regelmäßig, ob ein Server gesund ist.

Beispiele:

- Ping
  - \* TCP-Port erreichbar
  - \* HTTP-Status 200
  - \* eigene API-Healthchecks

Wenn ein Server nicht gesund ist → automatisch herausgenommen.

ASCII:

```
Server2 DOWN → Traffic nur an Server1 & Server3
```

---

# 5. TLS-Terminierung (SSL-Offloading)

Der Load Balancer entschlüsselt HTTPS und leitet intern HTTP weiter.

Vorteile:

- geringere Serverlast
  - \* zentrale Zertifikatsverwaltung
  - \* Content-based Routing möglich

Beispiel (Traefik, Nginx):

```
Client → HTTPS → LB → HTTP → Backend
```

## 6. Load Balancing in der Praxis (Beispiele)

### Beispiel 1: Webserver-Cluster

```
Nginx/Traefik LB → 3× Apache/PHP Server
```

### Beispiel 2: Mailserver High Availability

```
HAProxy (L4) → 2× Dovecot IMAP Server
```

### Beispiel 3: VPN-Cluster

```
HAProxy (L4) → 2× WireGuard Gateways
```

### Beispiel 4: Kubernetes

K8s nutzt:

- kube-proxy
  - \* ingress controller (Traefik, Nginx)
  - \* LoadBalancer Services

## 7. Load Balancer vs. Reverse Proxy

Load Balancer:

- verteilt Traffic
  - \* kann L4 oder L7 sein

Reverse Proxy:

- immer L7
  - \* nimmt Anfragen entgegen und leitet an Backend weiter

\* z. B. Traefik, Nginx, Apache mod\_proxy

Viele Produkte kombinieren beides.

---

## 8. High Availability (HA)

Oft nutzt man:

- VRRP (Keepalived)
  - \* Heartbeat
  - \* Pacemaker

ASCII:

```
VIP (Virtuelle IP)
  ↓
[LB1] <-> [LB2]
```

Wenn LB1 ausfällt → LB2 übernimmt.

---

## Zusammenfassung

- Load Balancing verteilt Anfragen auf mehrere Server
  - L4 = schnelles Routing nach IP/Port
    - \* L7 = Routing nach Inhalten (URLs, Header)
    - \* Health Checks prüfen Serverzustand
    - \* TLS-Offloading erleichtert Verwaltung
    - \* in modernen Architekturen unverzichtbar
    - \* Docker, Kubernetes, Cloud nutzen fast immer L7 Load Balancer

From: <http://wiki.nctl.de/dokuwiki/> - `Veni. Vidi. sudo rm -rf / vici.`

Permanent link: [http://wiki.nctl.de/dokuwiki/doku.php?id=it-themen:grundlagen:netzwerkdienste:load\\_balancing&rev=1764840339](http://wiki.nctl.de/dokuwiki/doku.php?id=it-themen:grundlagen:netzwerkdienste:load_balancing&rev=1764840339)

Last update: **04.12.2025 10:25**

